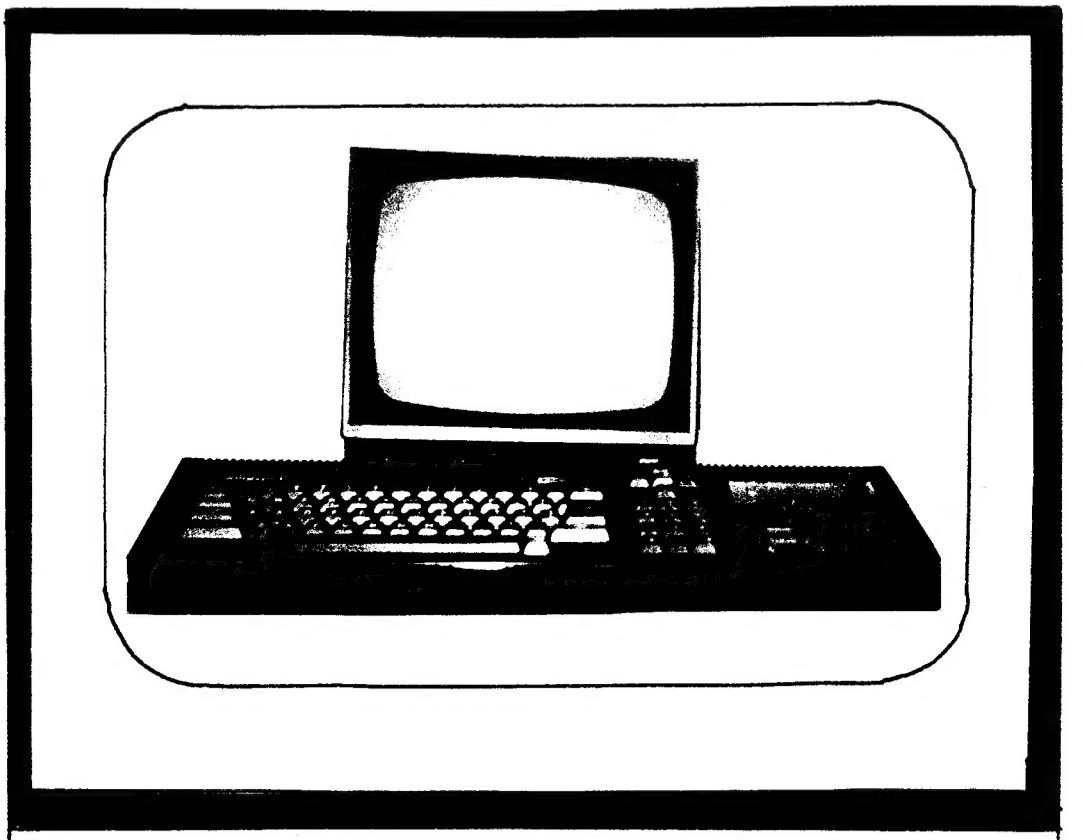
PRINT—OUT

Issue Two

Price 70p



BY Thomas Defoe, Mark Gearing and Jonathan Haddock

Including — Basic & M/C Programs

NAME AND ADDRESS STORER - Simple database

MATH'S MASTER PLUS - Homebrew Software

COMPETITION - Win Arnor's MAXAM

Miscellaneous

Features

Reviews

Programming

Once again we would like to take this opportunity to thank MR Gearing and Black Horse Agencies, Januarys, for their continued support in the use of their photocopier and typewriter in this issue.

EDITORIAL

WELCOME TO ISSUE TWO OF PRINT-OUT.

We were delighted by the support that was offered by many of our readers and the kind letters that we received.

The number of copies sold for Issue One was far beyond all our expectations, & we hope that Issue Two will see an increase in the number of readers.

We were also very pleased with the large number of questionnaires that were returned to us. We have tried to include as many of your ideas as space permits.

As a direct result of your questionnaires we have included in this issue :- A competition to win MAXAM, a tips page, many more programs and tutorials, a letters page and homebrew software. We have also changed our type-style and many other ideas are being considered for Issue Three.

Many of you have asked us to tell you the date when the next magazine will be out. We were reluctant to do this in case deadlines could not be met. However, we can tell you that Issue Three will be available in late December or early January. If you wish to send us £1·10 we will send you Issue 3 as soon as it is ready.

Others have asked if there are other ways to pay and the answers are shown here.

a) SUBSCRIPTIONS - Unless you live outside England there are NO subscription rates.
b) COST - The cost of the magazine is one of the following :- 70p + SAE (26p) or £1·10 (which is inclusive of P+P).

We hope that you enjoy Issue Two and that you will tell your friends about Print-Out. Please feel free to drop us a line to do with anything concerning the CPC. We try to answer all the letters we receive, but please don't expect an immediate answer.

UNTIL NEXT TIME. HAPPY READING !!!

COMPETITION -WIN MAXAM

Basic for BEGINNERS

At the end of the last issue we had a program which cleared the screen and printed a welcome message. This program is listed below.

```
1Ø CLS
2Ø PRINT "Hello ";
25 PRINT "Friend"
3Ø PRINT
4Ø PRINT "I am your Amstrad"
```

RENUM - By using this command we can renumber all the line numbers so they are multiples of ten.

AUTO - This command should be typed in before you start any programming and it will automatically print a line number, which is a multiple of ten, every time you press ENTER.

NEW - So that we can program a new program, type NEW and it will clear the memory of any BASIC programs.

Our next program will clear the screen and print a message asking for a person's name. We already know how to clear the screen and the command INPUT is used to ask the user what he or she wishes to do. The program below clears the screen and waits for the user's input.

```
1Ø CLS
2Ø INPUT name$
```

In this program 'name$' is a variable and the $ sign shows that it is a string variable as oppose to a numeric variable. A variable is a place where bits of information can be stored before they are used again. On the Amstrad we can have an almost infinite variety of variables, in fact, as large as the users imagination! The program above clears the screen and prints a question mark (?) and waits for the user to enter something that is followed by ENTER. It then stores the user's entry in name$. However, just printing a question mark and expecting someone to know exactly what to do isn't very helpful and so we can also print a piece of text using INPUT. Change line 20 to the one shown below.

```
2Ø INPUT "What is your name";name$
```

When you RUN the program now the screen will clear and the message, 'What is your name?' will appear on the screen and the computer will wait for you to enter something. If you now add line 3Ø, the computer will ask you for your name and then say hello to you.

```
3Ø PRINT "Hello ";name$
```

This shows that you can use variables with PRINT. The semi-colon is there so that the next piece of text to be printed will be on the same line. In this case the next piece of text is stored in the variable name$. If you now change line 30 as shown and add line 4Ø it will print a bit more afterwards.

```
3Ø PRINT "Hello ";name$;
4Ø PRINT " I am your Amstrad"
```

Another useful feature of Amstrad BASIC is its UPPER$ command. This converts a string so that it is written entirely in capital letters. Add line 25 and try it.

```
25 name$=UPPER$(name$)
```

This should cause your name to be written in capital letters. The opposite of UPPER$ is LOWER$ and it works in exactly the same way. Try it and see!

That was inputting text but what about inputting numbers. These are handled in exactly the same way except that you have to put a numerical variable at the end of the INPUT statement as opposed to a string variable. The program below will clear the screen and ask you for two numbers.

```
10 CLS
20 INPUT "Enter number 1",number1
30 INPUT "Enter number 2",number2
```

Note that by using a comma after the input text we have suppressed the question mark that appeared when there was a semi-colon after it. The next part of the program will find the average of them and print the answer. Add the following lines.

```
40 answer=(number1+number2)/2
50 PRINT "The answer is";answer
```

Note that there is no space between is and the speech mark (") and so you would have thought that the number would have followed straight on. However, all numbers on the Amstrad have a leading and trailing space. All that line 40 does is do the arithmetic. Anybody who knows how to calculate averages should be able to follow what it is doing. The answer is stored in the variable 'answer' which is printed in line 50. To print out the sum as well as the answer change these two lines.

```
50 PRINT "The average of";number1;
60 PRINT "and";number2;"is";answer
```

So that this program is really friendly we can add a few lines at the beginning which ask for your name and prints an introduction. The bulk of this you have already seen.

```
10 CLS
20 INPUT "What is your name";name$
30 PRINT "Hello ";name$;
40 PRINT " I am your Amstrad and"
50 PRINT "I am going to calculate averages"
60 INPUT "Enter number 1 ",number1
70 INPUT "Enter number 2 ",number2
80 answer=(number1+number2)/2
90 PRINT "The average of";number1;
100 PRINT "and";number2;"is";answer
```

The beauty of using variables is that you can change a program very simply and without having to alter the bulk of it. For example, to change the program so that it calculates the sum of two numbers, change the following lines.

```
50 PRINT "I am going to calculate totals"
80 answer=number1+number2
90 PRINT "The total of";number1;
```

Here is a brief summary of the new commands covered in this issue.
RENUM - Renumbers a program, with lines in steps of ten.
AUTO - Automatically prints line numbers when ENTER is pressed.
NEW - Clears the computer's memory of BASIC.
INPUT - Gets a piece of information from the user.
UPPER$ - Converts a string into capital letters.
LOWER$ - Converts a string into lower-case letters.

In this column, we give your thoughts an airing, try to solve your problems and give you the chance to speak to other CPC owners. Don't hesitate to write to us at the usual address.

Many thanks for the program tape and booklet. It all looks very interesting and extremely helpful as I am very much a novice at computing but am eager to learn. I really think the magazine is far better than Amstrad Action. I am looking forward to Issue 2 (even my wife enjoyed it!)

R.M. DUGGAN
HASTINGS

P-O : It's always nice to start off with pleasant things but this must take some beating. We all hope you enjoy Issue 2 as much.

This fanzine appealed to me because of its serious side, and as the Firmware Manual is no longer available I hope that you will be covering this fact too. I would like to see a toolkit without RSXs regarding sprite printing, over and under background with collision check and scrolling backgrounds with new scenery as is done in professional games, if possible. I'd like to see in future - hacking, break point routines, a tutorial on artifical intelligence and how to digitise sound.

S. MESSINA
HEYWOOD

P-O : In answer to your questions :-
Yes, we are going to cover the firmware calls and if anyone knows of a shop which has got firmware manuals still in stock, please tell us !!!
A toolkit, such as you request is not possible in this magazine due to space. However, there are some machine code routines which you may be able to adapt to suit your purposes in this isue. If anyone else has any routines please send them in as I am sure they will be appreciated. All of your other suggestions have been taken into account and if enough people show an interest, we may well print them in the future.

Many thanks for your reply to my letter and for my copy of Print-Out. May I congratulate you on a first class issue one. With regard to your requests for suggestions and comments may I offer the following :-
1. I found that quite often the printing disappeared under the binder.
2. Name the authors of articles, as it gives the reader an extra point of contact with the magazine.
3. Include really advanced levels of BASIC and M/C programming so as to attract the widest possible readership.
4. Local computer shops may have special offers which you could advertise. You may get some advertising revenue from this or perhaps a donation to offer as a prize.

R.J. TAYLOR
HARLOW

P-O : In reply to your suggestions :-
1. We have expanded our margins in an attempt to eliminate this.
2. We understand what you are saying, and in fact would do that, but as each of us contributes to each article it is not very sensible.
3. We have tried to include a wide range of articles, for the beginner and the advanced user. However, if there is anything specific that you would like us to include then please tell us.
4. If you know of any shops that might be willing to advertise in this magazine then please tell us. It doesn't matter where they are !!!

Program 1

Name and Address Storer — an example database listing

Instructions

This program is designed to be an example of a simple database. Unlike a database, however, the fields are already set and it is designed for the express purpose of holding names and addresses. The program allows you to hold up to 1000 names, addresses and telephone numbers. We have tried to make it as extensive and complete as possible, but due to its size there will be omissions. The program gives you eight main options and they are shown below :-

LOAD a file - This allows you to load a previously saved file into the program for editing, viewing, etc.

SAVE a file - This allows you to save a file which has been created or edited for future use. When ready it is reloaded using the LOAD a file command. By using the save command you do not lose the information in the program before the SAVE was issued. By using option 8 - QUIT program - you can clear all the data from the program and then restart.

EDIT a file - This enables you to change any entry in the file, either because of a mistake or a change of address, etc. You can change any entry either totally or in bits (eg. just the phone number). The program prints up many on-screen messages to help you enter the correct information.

VIEW a file - This displays the entire file alphabetically (presuming you have used option 5 - SORT a file) or in the order they were entered. At any time you can return to the main menu to make another choice.

SORT a file - This routine sorts the file alphabetically (using the name). There are one or two small problems with this routine. The biggest being the if you have two people with identical surnames it will come up with an error message. One way round this is to enter names in the following way :- SURNAME, INITIALS (eg. BLOGGS, J.) as long as you enter them in a consistent manner the program should be alright.

INI

FIND entry - This allows you to search through a file until it finds an entry. The computer will ask you for either the name, phone number or address (only the first line) and then search the file until it finds the entry. If it does not exist then the computer will tell you.

```
 10 REM Name and Address Storer
 20 REM (c) 1989, Thomas Defoe
 30 REM
 40 GOSUB 280:REM Initialisation
 50 GOTO 100:REM Main Menu
 60 MODE 2:PRINT "Do you want or restart (Y/N) ?"
 70 a$=INKEY$:IF a$="" THEN 70
 80 IF UPPER$(a$)="Y" THEN RUN
 90 MODE 2:END
100 REM Main Menu
110 CLS:LOCATE 34,3:PRINT "MAIN MENU"
120 LOCATE 27,5:PRINT "Please enter your choice"
130 LOCATE 30,7:PRINT "1) LOAD a file"
140 LOCATE 30,8:PRINT "2) SAVE a file"
150 LOCATE 30,9:PRINT "3) EDIT a file"
160 LOCATE 30,10:PRINT "4) VIEW a file"
170 LOCATE 30,11:PRINT "5) SORT a file"
180 LOCATE 30,12:PRINT "6) FIND entry"
190 LOCATE 30,13:PRINT "7) CREATE file"
200 LOCATE 30,14:PRINT "8) QUIT program"
210 know$="12345678":where=0
220 a$=INKEY$:IF a$="" THEN 220
230 where=INSTR(know,a$)
240 IF where=0 THEN 220
250 ON where GOSUB 370,520,670,1120,1850,1280,1650,60
260 GOTO110
270 END
280 REM Initialisation
290 MODE 2:INK 0,0:INK 1,24:PAPER 0:PEN 1:BORDER 7
300 MOVE 0,0:DRAW 639,399:DRAW 639,0:DRAW 0,0
310 WINDOW #0,2,79,4,20:WINDOW #1,2,79,21,4:PEN #1,0
320 PAPER #1,1:CLS #1:WINDOW #2,2,79,2,3
330 LOCATE #2,27,1:PRINT #2,"Name and Address Storer"
340 DIM name$(1000):DIM phone(1000):DIM address$(1000)
350 number=0
360 RETURN
370 REM Load a file
380 CLS:LOCATE 20,6:INPUT "Please enter filename :-  ",file$
390 IF LEN(file$)>6 THEN 380
400 OPENIN file$
410 INPUT #9,number
420 FOR i=1 TO number
430 INPUT #9,name$(i)
440 INPUT #9,phone(i)
450 FOR j=1 TO 4
460 INPUT #9,address$(i,j)
470 NEXT j
480 NEXT i
490 CLOSEIN
500 CLS
```

```
 510 RETURN
 520 REM Save a file
 530 CLS:LOCATE 20,6:INPUT "Please enter filename :-  ",file$
 540 IF LEN(file$)>6 THEN 530
 550 OPENOUT file$
 560 PRINT #9,number
 570 FOR i=1 TO number
 580 PRINT #9,name$(i)
 590 PRINT #9,phone(i)
 600 FOR j=1 TO 4
 610 PRINT #9 ,address$(i,j)
 620 NEXT j
 630 NEXT i
 640 CLOSEOUT
 650 CLS
 660 RETURN
 670 REM Edit a file
 680 CLS:PRINT "What is the person's name ";
 690 INPUT ed$
 700    ed$=UPPER$(ed$)
 710 FOR i=1 TO number
 720 IF ed$=UPPER$(name$(i)) THEN GOTO 760
 730 NEXT i
 740 PRINT "There is no-one called ";UPPER$(ed$)
 750 FOR T=1 TO 1500:NEXT t:RETURN
 760 REM
 770 LOCATE #1,4,2:PRINT "Person number :-  ";i
 780 LOCATE 20,4:PRINT "Name :- "; name$(i)
 790 LOCATE 20,5:PRINT "Phone :- ";phone(i)
 800 LOCATE 20,6:PRINT "Address :- ";
 810 FOR j=1 TO 4
 820 LOCATE 20,6+j:PRINT address$(i,j)
 830 NEXT i
 840 LOCATE 20,13:PRINT "Is the NAME right (Y/N) ?"
 850 a$=INKEY$:IFa$="" THEN 850
 860 IF UPPER$(a$)="Y" THEN 880
 870 LOCATE 20,14:PRINT "Enter correct name :- ";name$(i)
 880 LOCATE 20,13:PRINT SPACE$(70):LOCATE 20,14:PRINT SPACE$(70)
 890 LOCATE 20,13:PRINT "Is the PHONE number right (Y/N) ?"
 900 a$= INKEY$:IF a$="" THEN 900
 910 IF UPPER$(a$)="Y" THEN 940
 920 LOCATE 20,14:INPUT "Enter correct number :- ",phone(i)
 930 LOCATE 20,13:PRINT SPACE$(70):LOCATE 20,14:PRINT SPACE$(70)
 940 LOCATE 20,13:PRINT "Is the ADRESS right (Y/N) ?     "
 950 a$=INKEY$:IF a$="" THEN 950
 960 IF UPPER$(a$)="Y" THEN 1030
 970 FOR j=1 TO 4
 980 LOCATE 20,13+j:PRINT "Enter correct line";j;
 990 INPUT "of address :- ",address$(i,j)
1000 LOCATE 20,13:PRINT SPACE$(70):LOCATE 20,14:PRINT SPACE$(70)
```

```
1010 NEXT j
1020 CLS
1030 CLS:LOCATE 20,4:PRINT "Name :- ";name$(i)
1040 LOCATE 20,5:PRINT "Phone :- ";phone(i)
1050 LOCATE 20,6:PRINT "Address :- ";
1060 FOR j=1 TO 4
1070 LOCATE 24,6+j:PRINT address$(i,j)
1080 NEXT i
1090 LOCATE 20,12:PRINT "Press any key to continue"
1100 a$=INKEY$:IF A$="" THEN 1100
1110 RETURN
1120 REM View a file
1130 FOR i=1 TO number
1140 CLS
1150 LOCATE 6,3:PRINT "Person :- ";i
1160 LOCATE 20,4:PRINT "Name :- ";name$(i)
1170 LOCATE 20,5:PRINT "Number :- ";number(i)
1180 LOCATE 20,6:PRINT "Address :-"
1190 FOR j=1 TO 4
1200 LOCATE 24,6+j:PRINT address$(i,j)
1210 NEXTj
1220 LOCATE 20,12:PRINT "Press any key to continue"
1230 LOCATE 20,13:PRINT "or SPACE to return to menu"
1240 a$=INKEY$:IF a$="" THEN 1240
1250 IF a$=" " THEN RETURN
1260 NEXT i
1270 RETURN
1280 REM Fnd Entry
1290 CLS:LOCATE 20,4:PRINT "Select one from the following :-"
1300 LOCATE 26,6:PRINT "1) Name"
1310 LOCATE 26,7:PRINT "2) Phone"
1320 LOCATE 26,8:PRINT "3) Address"
1330 know$="123":where=0
1340 en=1
1350 a$=INKEY$:IF a$="" THEN 1350
1360 where=INSTR(know$,a$)
1370 IF where=0 THEN 1350
1380 ON where GOSUB 1490,1550,1600
1390 IF en=0 THEN RETURN
1400 CLS:LOCATE 20,7:PRINT "Name :- ";name$(i)
1410 LOCATE 20,8:PRINT "Phone :- ";phone(i)
1420 LOCATE 20,9:PRINT "Address :-"
1430 FOR j=1 TO 4
1440 LOCATE 24,9+j:PRINT address$(i,j)
1450 NEXT j
1460 LOCATE 20,15:PRINT "Press any key to continue"
1470 a$=INKEY$:IF a$="" THEN 1470
1480 RETURN
1490 CLS:LOCATE 20,4:INPUT "What is the name ";find$
1500 find$=UPPER$(find$)
```

```
1510 FOR i=1 TO, number
1520 IF UPPER$(name$(i))=find$ THEN RETURN
1530 NEXT i
1540 en=0:RETURN
1550 CLS:LOCATE 20,4:INPUT "What is the number ";find
1560 FOR i=1 TO number
1570 IF phone(i)=find THEN RETURN
1580 NEXT i
1590 en=0:RETURN
1600 CLS:LOCATE 20,4:INPUT "What is the address ";find$
1610 FOR i=1 TO number
1630 IF UPPER$(address$(i,j))=UPPER$(find$) THEN RETURN
1640 en=0:RETURN
1650 REM Create a file
1660 number=number+1
1670 LOCATE #1,4,2:PRINT #1,"Number of entries is :-";number
1680 field$="NAME   ":GOSUB 2070
1690 CLS:LOCATE 4,5:PRINT "What is the name of the person";number;
1700 INPUT name$(number)
1710 IF UPPER$(name$(number))="*END" THEN number=number-1:CLS #1:RETURN
1720 field$="PHONE  ":GOSUB 2070
1730 LOCATE 4,7:PRINT"What is the telephone number ";
1740 INPUT phone(number)
1750 field$="ADDRESS":GOSUB 2070
1760 FOR i=1 TO 4
1770 LOCATE 4,8+i:PRINT "What is line";i;" of the address";
1780 INPUT address$(nmuber,i)
1790 NEXT i
1800 field$="CHECK  ":GOSUB 2070
1810 LOCATE 4,15:PRINT "Is all this information correct ?"
1820 a$=INKEY$:IF a$="" THEN 1820
1830 IF UPPER$(a$)="N" THEN GOTO 1680
1840 number=number+1:GOTO 1670
1850 REM Bubble Sort
1860 FOR i=1 TO number
1870 name$(i)=UPPER$(name$(i))
1880 NEXT i
1890 FOR k=1 TO number
1900 FOR i=1 TO number
1910 one=i:two=i+1
1920 IF two=number+1 THEN GOTO 1950
1930 GOSUB 1970
1940 NEXT i
1950 NEXT k
1960 RETURN
1970 REM Sort
1980 j=i
1990 IF j>15 THEN PRINT "Please change name":FOR t=1 TO 3000:NEXT t:RETURN
2000 a$=MID$(name$(one),j,1)
2010 a$=MID$(name$(two),j,1)
```

```
2020 a=ASC(a$)
2030 b=ASC(b$)
2040 IF a=b THEN j=j+1:GOTO 1990
2050 IF a>b THEN GOSUB 2090
2060 RETURN
2070 LOCATE #1,4,3:PRINT #1,"Entry being made :- ";field$
2080 RETURN
2090 REM Swap
2100 hold$=name$(one):phone(one)=phone(two):phone(two)=hold$
2110 pold=phone(one):phone(one)=phnoe(two):phone(two)=pold
2120 FOR z=1 TO 4
2130 add$=address$(one,z):address$(one,z)=address$(two,z)
2140 address$(two,z)=add$
2150 NEXT z
2160 RETURN
```

OOPS!

Line 211Ø should read :-

```
211Ø pold=phone(one):phone(one)=phone(two):phone(two)=pold
```

and 117Ø

```
117Ø LOCATE 2Ø,5:PRINT "Number :- ";phone(i)
```

CONT.

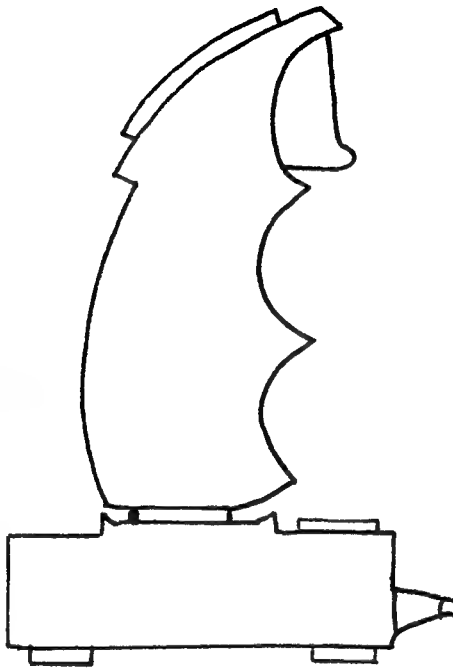
CREATE file - This allows you to enter as many names, addresses and telephone numbers as you want (upto a maximum of 1000). To end this option type - *END - instead of a name and you will be returned to the main menu. After you have entered all the information on one person (some can be left blank) you will be asked whether it is correct. If it is you will have a chance to enter information for the next entry otherwise you will be able to redo the same entry. The program allows you to enter 1 line of names, 1 line of phone numbers and 4 lines of address per person.

QUIT program - This self explanatory, upon selecting this option, you will be asked if you wish to start again. If you don't the program will end. If you do restart the program will recommence with all data removed. Thus it can be used as a short-cut instead of reloading the program.

The program has on-screen messages through-out which we hope will help you not to become lost or confused whilst using it. A slightly modified version is used to store our information as to which orders have been despatched, printed, recorded etc. and we find that it is a great help and reduces the number of sheets of paper with addresses that are lying around.

GAMES REVIEWS

BATMAN *from*



Review

This is another budget releases from The Hit Squad and costs £2·99 on tape only. This is Batman's second release - it was first released as a full price game in 1986 by Ocean. It is also the predecessor of the excellent Head Over Heals (again from Ocean) and is indeed written by the same people, Jon Ritman and Bernie Drummond. Batman should not be confused with 'Batman - The Caped Crusader' or the movie tie-in which will no doubt be quick to appear.

As you can guess, you play Batman who, with the aid of his batboots, batbag, bat-thruster and batbelt (and anything else he can lay his hands on!), must save Robin, who is the prisoner of the Joker and the Riddler. The only way to rescue Robin is to find the various pieces of the batcraft which lie scattered around the maze beneath the Batlair. Once the batcraft has been assembled, Batman needs to find the launching-pad from which he can fire up the engines and go in search of Robin.

As usual, you are given the choice of using the standard keys or redefining your own. You can adjust the level of sound in the game to suit all tastes and also adjust the sensitivity. However, the sensitivity menu was poorly explained and seemed to do very little to the game. All of these options are controlled from a series of menus. Owners of Head Over Heels, Matchday or Matchday II will already be familiar with this style of menu.

As might be expected, the game is viewed in 3D perspective and movement is also in 3D. This adds greatly to the game as detailed and complicated moves can be executed. The graphics are similar to Head Over Heels & as usual are very detailed and beautifully animated. The sound is not outstanding, just a few sound effects, but the game does include several good tunes at the beginning and end. The playing area is large and most rooms contain a puzzle or problem which must be solved before access can be made into other rooms. These puzzles are probably the game's greatest asset - after all, no matter how beautifully the game looks or sounds, it is the problems that make it. However, once you know what order to place the blocks in, it is relatively simple to progress. Batman is slightly slow in that because the man moves slowly - the game moves slightly slower than I would have liked. Having said that, Batman is still a truely first- class game even though it is 3 years old. Excellent !!!

Mark

I found the menu system (especially defining the keys) very complicated and I am sure that it could have been done better. The instructions were also rather sketchy on this point but this was probably due to lack of space. The graphics were good as were the sound effects, but I missed the tune during the actual game. I found the game quite enjoyable. Overall an average game, but at this price - whose complaining.

78%

Jon

Batman is an excellent game with a well thought out layout and plot. The graphicssare extremelv good and the sound was acceptable, if not brilliant. The skill level on this game is just right and I think that it will take a very long time to complete.

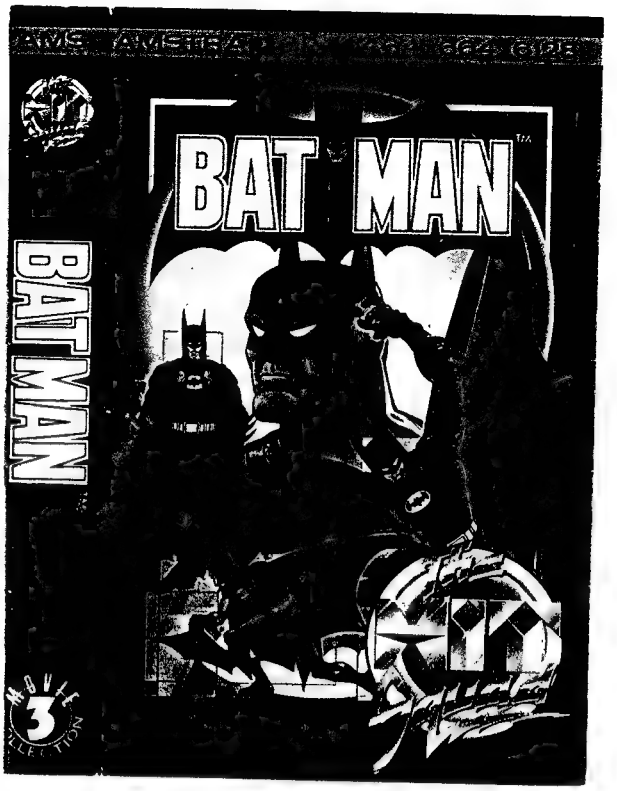
83%

Tom

As a person who already owns Head Over Heels, I was looking forward to reviewing Batman. I was not disappointed. The graphics are of the usual high standards and the sound effects are well done. This game will suit everyone. The arcade fan will find something in it, the strategy planner will enjoy it and the normal games-player will have a field day!! This game is probably as close to an arcade-adventure as we will ever get. It is without doubt, the gameplay which makes this game something to remember. At no time did it become boring, and it's horribly addictive.

80%

Batman from the Hit Squad costs £2·99 on tape only.



COBRA FORCE

PLAYERS PREMIER

Review

This is another budget game which costs £2·99 (tape only) and is this time released by Players Premier. This game has no real,strong story-line - just blast and destroy !!!! However, from the instructions you do discover that you are flving an AH-1W Super Cobra that is armed with a standard M197 20mm machine gun and twenty or so Hellfire armour piercing missiles. Your aim is both complex and difficult. Not only do you have to reach the end of the level but collect all the fuel drums and destroy all the defence installations as well as the 'enemies' awesome weapon' - then onto level 2.

When certain alien ships are destroyed, your helicopter gains extra weaponry or a replenished missile supply. The graphics are good and are extremely detailed as well as being colourful. The main fault with the game is the lack of sound. There are just a few feeble effects and there is no music/tune of any kind throughout the entire game. The other disappointment is the inability to redefine the keys. The instructions are also sadly lacking - by pressing the 'BOMB' key you can use a Smart bomb, what BOMB key ?

Information is clearly displayed and the scrolling is very smooth, and the explosions are well done. However, it is after all just another shoot-'em-up but at £2·99 it is probably one of the best. If you like traditional blasting games then this is for you.

Mark

I found Cobra Force very addictive, as I particularly enjoy fast action games. The graphics were excellent but the sound was poor - I would have liked a catchy tune! I think that it represents excellent value for money and rank it very highly indeed.

82

Jon

The graphics in Cobra Force were excellent and the screen layout was also good. I disliked the key layout and found the game far too hard. The lack of music also detracted from the game. Not a particularly good game but at this price you don't expect too much.

66

Tom

The graphics were exceptional, the gameplay was good, the controls were acceptable, the instructions were awful, the sound was abysmal and the music was non-existent. It all depends what you want from a game whether you like this. I did!

75

BASIC Graphics

COLOUR

In this,the second part of our guide to the Amstrad's graphics commands, we look at colours, text and various other commands which were not covered in the last issue.

Probably the most confusing part of Locomotive BASIC is the way in which colours are used and, of course, colours are VERY important. In each of the three screen MODEs, you can have a set number of colours displayed at any one time and these are shown in the table below. There is also a column headed 'SCREEN SIZE (Memory)' and this tells you how much memory one screen takes up in each of the MODEs. If you don't know about memory and 'K' you should read the section 'What is my Amstrad ?' in Issue 1.

MODE	NUMBER OF COLOURS	SCREEN SIZE (Memory)
0	16 simultaneously	16K
1	4 simultaneously	16K
2	2 simultaneously	16K

How, you may be asking, can MODE 0 have 16 colours available and still take up the same amount of memory as MODE 2 which has only two colours available? The answer is simple. In the different modes the memory is used in different ways, so 16K is still used to store ALL the screen data but this data changes according to the mode you are in.

Now, this simple answer is all very well but we really need to be a bit more specific about. In order to explain this I will need to remind you of some points that came up in Issue 1. Firstly, the screen is made up of a grid of pixels of size 640 x 200. In different modes you can address different numbers of pixels independently. In MODE 2 you can address all 640 x 200 (or 128,000) pixels separately, in MODE 1 you can control 320 x 200 (or 64,000) pixels and in MODE 0, 160 x 200 (or 32,000) pixels on their own. Of course the other pixels don't disappear but the horizontal pixels are lit up in groups. In MODE 0 they are lit in groups of 4, in MODE 1 in pairs and in MODE 2 on their own.

As we have already said, in MODE 2 you can address 128,000 pixels and as you have only two colours, the foreground and the background, the pixels can either be on or off. This information about the state of the pixels is easy to represent in Binary - 0s (off) and 1s (on). Therefore the data for one pixel can be stored in one bit and so a MODE 2 screen can be stored as 128,000 bits. This can be written as 16,000 bytes, as there are 8 bits in a byte, or 16 kilobytes (1000 bytes in a kilobyte). This is where the screen size 16 K is found.

That has only explained half the problem. We now know why a MODE 2 screen takes up 16K but why do MODE 0 and MODE 1 screens do the same ? We know already that MODE 1 screens have 64,000 separate pixels and this is the equivalent to 8K of MODE 2 data. However, in MODE 1 you have four colours and these cannot be all be stored in binary using just 0s and 1s as with MODE 2. Therefore to distinguish between four colours you need four combinations of 0s and 1s and they are 00, 01, 10 and 11.

As you will have realised you cannot store that information about MODE 1 colours in just one bit - but you need to bits. So for every pixel two bits are assigned and thus 128,000 bits are needed (64,000 x 2) and this is also 16K.

Likewise in MODE 0 you need to store information for 16 colours and so in binary these are represented as :- 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110 and 1111. As the horizontal pixels are lit in groups of four, you have 32,000 independent pixels and each of these needs four bits (or ½ byte) and so it takes up 128,000 bits or 16K.

So much for all the theory, but what colours can we use? The Amstrad has 27 colours available and they are :-

Number	Colour	Number	Colour
0	Black	14	Pastel Blue
1	Blue	15	Orange
2	Bright Blue	16	Pink
3	Red	17	Pastel Magenta
4	Magenta	18	Bright Green
5	Mauve	19	Sea Green
6	Bright Red	20	Bright Cyan
7	Purple	21	Lime Green
8	Bright Magenta	22	Pastel Green
9	Green	23	Pastel Cyan
10	Cyan	24	Bright Yellow
11	Sky Blue	25	Pastel Yellow
12	Yellow	26	Bright White
13	White		

The small program below will allow you to see them in all their colourful glory, unfortunately for green screen owners, you'll just see different shades of green.

```
10 REM Colour Printer
20 MODE 0:PAPER 0:BORDER 7
30 c1=0:c2=14:FOR j=c1 TO c2
40 INK j,j:NEXT j:INK 15,24
50 LOCATE 7,1:PEN 15:PRINT "Part 1"
60 FOR j=1 TO c2:FOR i=1 TO 40
70 MOVE i+(j*40),300:DRAW i+(j*40),350,j
80 NEXT i:NEXT j
90 IF c2=11 THEN GOTO 90
100 LOCATE 4,9:PEN 15:PRINT "Press anv kev"
110 a$=INKEY$:IF a$="" THEN 110
120 c2=11:FOR j=c1+1 TO c2
130 INK j,j+15
140 NEXT j
150 INK 0,0
160 MODE 0:LOCATE 7,1:PEN 14:PRINT "Part 2":GOTO 60
```

Earlier I mentioned foreground and background and now we will turn our attention to these as we have ways of setting the colour of both. The background is the main part of the screen that all letters, lines and patterns appear on and so is like a sheet of paper. The foreground is all of the bits that appear on the background like lines and letters and is similar to the pen.

PAINT POTS

Probably the easiest way of thinking of colours on the Amstrad is to imagine that in MODE 0 you are given 16 paint brushes, four in MODE 1 and two in MODE 2. You can dip each of these paint brushes into any of the 27 paint pots provided but are limited by the number of paint brushes that you are given.

To dip your paint brushes in a paint pot you use the command INK. You can then select the background colour by using PAPER and the foreground colour by using PEN.

INK - This is used to select the colour to be assigned to a certain pen or paper and is followed by two numbers. The first of these is the number of the paint brush that you are assigning a colour to, and the second is the number of the colour. So to set the ink in paint pot 1 to black you would use the command - INK 1,Ø.

PEN - This command is used to set the foreground colour and is followed by one number. This number is the number of the paint pot that is to be used and not the colour's number. So to set the foreground to yellow you would use the commands - INK 1,12:PEN 1.

PAPER - This is used to set the background colour and is similar in use to PEN. It is also followed by one number and this is also the number of the paint pot that it is to use. To set the background bright green, you would use the commands - INK Ø,2Ø:PAPER Ø

FLASHING INKS

To have flashing colours is really very simple. You just give the INK command an extra number which specifies what the second ink will be. For example, to set INK 1 to flash bright white then black use the command - INK 1,26,Ø. To vary the speed of flashing you use the command

SPEED INK - This is followed by two numbers. The first says how long the first colour should be on the screen in 1/50 second and the second number specifies the time the second colour should be on the screen. So to have a sudden stab of bright white followed by a long gap use the commands - INK 1,26,Ø:SPEED INK 1,50.

BORDER - This sets the colour of the border of the screen which acts rather like a picture frame. Unlike PEN and PAPER it uses the value of the colour rather than the value of an ink. So to set the border to red use - BORDER 3.

TEXT

You may remember that you could print and position text by using the LOCATE x,y command. However, for some purposes this may be a little inaccurate and so you can use TAG. After issueing the command TAG all text will be printed at the graphics cursor position complete with a few extra arrows. To remove the extra arrows you must end the PRINTing line with an ;. So to print the word Hello at 1ØØ,2ØØ use the commands

```
TAG:MOVE 1ØØ,2ØØ:PRINT "Hello";:TAGOFF
```

The command TAGOFF simply tells the computer to resume printing text at the text cursor as opposed to the graphics cursor.

That rounds off this issue's section on BASIC Graphics but there are also sections on Pallette Swapping and Multi-coloured Graphics. So until the next issue, keep experimenting !!!

NB : This program appeared on the Program Cassette for Issue 1 and was designed to be published along with the Language Tester program in our first issue. However, limitations in space forced us to omit it and it has been adapted to run as a stand-alone program.

CHARACTER DESIGNER

This program will allow you to create and design characters for use in your own programs or, as it was designed to, create foreign letters for use with the Language Tester program in Issue 1. The program offers you three choices and they are :-

CHOOSE Character - This allows you to see what character you are about to edit. It defaults to ASCII 65, the letter A.

EDIT Character - This allows you to edit the character chosen & you control the cursor by using the cursor keys. SPACE changes the pixel and ENTER returns you to the main menu. In the box labelled 'IMAGE' the character is shown as it would appear when printed for real.

```
 10 REM **********************
 20 REM * Character Designer *
 30 REM **********************
 40 MODE 1:INK 0,0:INK 1,24:INK 2,16:INK 3,9:PEN 1:PAPER 0:BORDER 7
 50 MOVE 0,0:DRAW 0,399,1:DRAW 639,399:DRAW 639,0:DRAW 0,0
 60 MOVE 12,252:DRAW 12,386:DRAW 146,386:DRAW 146,252:DRAW 12,252
 70 LOCATE 15,3:PRINT "Character Designer"
 80 MOVE 220,351:DRAW 512,351:MOVE 200,0:DRAW 200,399
 90 WINDOW #1,2,9,29
100 WINDOW #2,2,11,11,12
110 WINDOW #3,2,11,16,23
120 WINDOW #4,14,39,5,16
130 WINDOW #5,14,39,11,23
140 char=65:endchar=255
150 DIM grid(9,9)
160 DIM tot(8)
170 lin=20
180 GOSUB 230
190 MOVE 0,200:DRAW 200,200
200 LOCATE 4,15:PRINT "IMAGE"
210 GOSUB 290:REM Main Menu
220 END
230 REM ********************
240 REM * Print Character *
250 REM ********************
260 LOCATE 2,11:PRINT "ASCII -";char
270 LOCATE 2,12:PRINT "Char  -";CHR$(char)
280 RETURN
290 REM *************
300 REM * Main Menu *
310 REM *************
320 PEN 1:LOCATE 16,6:PRINT "1) Choose Character"
330 LOCATE 16,7:PRINT "2) Edit Character"
340 LOCATE 16,8:PRINT "3) Calculate Values"
350 know$="123"
360 in$=INKEY$:IF in$="" THEN 360
370 where=INSTR(know$,in$)
```

PAGE 2 (Program continued)

```
380 IF where=0 then 360
390 ON where GOSUB 410,500,980
400 GOTO 290
410 REM ********************
420 REM * Choose Character *
430 REM ********************
440 LOCATE 16,6:PEN 3:PRINT "1) Choose Character"
450 CLS #5:PEN #5,1:LOCATE #5,2,2:INPUT #5,"Enter ASCII code:-",char
460 IF char<32 OR char>255 THEN GOTO 450
470 PEN 1
480 GOSUB 230:CLS #5
490 RETURN
500 REM ******************
510 REM * Edit Character *
520 REM ******************
530 PEN 3:LOCATE 16,7:PRINT "2) Edit Character":PEN 1
540 CLS #1:CLS #3
550 x=2:y=2ox=x:oy=y
560 FOR i=1 TO 9:FOR j=1 TO 9
570 grid(i,j)=0
580 NEXT j,i
590 LOCATE x,y:PEN 1:PRINT CHR$(143)
600 IF INKEY(0)<>-1 AND y>2 THEN y=y-1
610 IF INKEY(2)<>-1 AND y<9 THEN y=y+1
620 IF INKEY(8)<>-1 AND x>2 THEN x=x-1
630 IF INKEY(1)<>-1 AND x<9 THEN x=x+1
640 IF INKEY(47)<>-1 THEN GOSUB 730
650 IF INKEY(18)<>-1 THEN RETURN
660 IF ox=x AND oy=y THEN GOTO 600
670 LOCATE x,y:PEN 1:PRINT CHR$(143)
680 IF grid(ox,oy)=1 THEN PEN 3 ELSE PEN 0
690 LOCATE ox,oy:PRINT CHR$(143)
700 ox=x:oy=y
710 FOR i=1 TO 30:NEXT i
720 GOTO 600
730 REM ****************
740 REM * Change Pixel *
750 REM ****************
760 IF grid(x,y)=0 THEN GOSUB 800:RETURN
770 IF grid(x,y)=1 THEN GOSUB 890:RETURN
780 WHILE INKEY(47)<>-1:WEND
790 RETURN
800 REM ******
810 REM * ON *
820 REM ******
830 grid(x,y)=1
840 PLOT 60+(2*x),100-(2*y),1
850 WHILE INKEY(47)<>-1:WEND
860 IF TEST(60+(2*x),100(2*y))<>-1 THEN GOTO 840
870 ox=x:oy=Y
880 RETURN
890 REM *******
900 REM * OFF *
910 REM *******
```

PAGE 3 (Program continued)

```
 920 grid(x,y)=0
 930 PLOT 60+(2*x),100-(2*y),0
 940 HILE INKEY(47)<>-1:WEND
 950 IF TEST(60+(2*x),100-(2*y))<>0 THEN GOTO 930
 960 ox=xoy=y
 970 RETURN
 980 REM ************
 990 REM * Calculate *
1000 REM ************
1010 PEN 3:LOCATE 16,8:PRINT "3) Calculate Values":PEN 1
1020 FOR i=1 TO 8
1030 tot(i)=0
1040 NEXT i
1050 FOR row=2 TO 9
1060 RESTORE 1220
1070 FOR column=2 TO 9
1080 READ worth
1090 IF grid(column,row)=1 THEN tot(row-1)=tot(row-1)+worth
1100 NEXT column
1110 NEXT row
1120 LOCATE 16,11:PRINT "The values for character"
1130 LOCATE 16,12:PRINT char;"are :-"
1140 FOR i=1 TO 8
1150 LOCATE 16,i+13
1160 PRINT tot(i)
1170 NEXT
1180 PRINT:LOCATE 15,23:PRINT "Press any key to continue"
1190 a$=INKEY$:IF a$="" THEN 1190
1200 CLS #5
1210 RETURN
1220 DATA 128,64,32,16,8,4,2,1
```

The third option is :-

CALCULATE Values - When you are satisfied with your design, by pressing '3' a list of numbers will appear and these will need to be written down. When you want to use the new symbols type in BASIC, or in your own programs

```
SYMBOL AFTER 32
SYMBOL n,list of numbers
```

Where n stands for the character's ASCII code and in place of list of numbers you put the numbers that you noted down earlier. Here is an example of a modified letter 'A'.

```
SYMBOL 65,252,30,243,51,255,51,243,Ø
```

For details on how to put these new symbols into the Language Tester program see Issue 1.

One or two errors have crept in and they are :-

```
55Ø x=2:y=2:ox=x:oy=y
86Ø IF TEST(6Ø+(2*x),1ØØ-(2*y))<>-1 THEN GOTO 84Ø
96Ø ox=x:oy=y
```

QUESTIONNAIRE RESULTS

The response to our prize questionnaire in Issue 1 was suprisingly good and so we decided to print the results for all to see and they are shown below.

1) What type of Amstrad have you got ? CPC 464 - 50% CPC 6128 - 50%

2) What other computers, if any, have you got ? The most common answer was a Videogenie but Amstrad PCs, Amigas and Atari STs also featured.

3) What is your main use for your computer ?

Word processing	- 40%
Programming	- 50%
Games playing	- 5%
Other uses	- 5%

4) What program/game/utility do you use most ? Most people seemed to use a word processor of sorts, Mini Office II or some of their own utilities. The most popular word processor was Protext (40%) with Brunword, Amsword and Tasword each having a 20% share.

5) What hardware, if any, do you own ? The most popular item was a printer (50%) of various makes, then came an FD-1 Second Drive (30%) followed by a DD-1 First Drive (20%) for the CPC 464. It was suprising the number of people who had bought an FD-1 for their CPC 6128 as a second drive as opposed to a larger drive. Also featuring were ROM Boards, mice, light pens, speech synthesizers, clocks and, of course, the Multiface II+.

6) How long have you had your Amstrad ? The average length of time appears to be 3 years although replies varied from 5 to ½ a year.

7) & 8) As these were so varied and diverse we have decided not to print them but rest assured, they have all been taken into consideration. In general though, you seemed to like the magazine as a whole but did have one or two minor criticisms and we have tried to solve them. Whilst we are on the subject of complaints, we thought we would say how delighted we were to receive so many letters concerning the discontinuation of the Firmware Manual. We will pass all the letters on to Amstrad PLC and we hope that they and many others will change their minds. So if you've got something to say, say it through us.

9) Do you want more or less or the same of the following articles ?

PROGRAMS	More - 80%	Same - 10%	Less - 10%
REVIEWS	More - 50%	Same - 15%	Less - 35%
ARTICLES	More - 50%	Same - 35%	Less - 15%
HARDWARE	More - 25%	Same - 50%	Less - 25%
BASIC	More - 75%	Same - 20%	Less - 5%
MACHINE CODE	More - 80%	Same - 10%	Less - 10%

10) What thing(s) would you like us to include in future issues ? Again these were very varied but we have printed some ideas below and the most popular ones have been included in this issue. They are competitions, letters, tips page, bank manager, sound, poking and hacking and serious software reviews.

11) How did you hear about PRINT-OUT ? Of course the answer to this was almost unanimous. AMSTRAD ACTION. It seems only fitting that we should use this opportunity to thank Amstrad Action for all their help and publicity.

The Winner of our competition is S.J.Cookson from Hesketh Bank.

MACHINE CODE

First of all, an apology. Many thanks to all those people who wrote in to point out the error in the last issue's Machine Code section. This refers to the Zero flag. I think that the best way to clear this problem up is to let someone who spotted the error, explain. So here is what BOB TAYLOR of Harlow has to say on it :- 'On page 33 in lines 20 to 22 it is mistakenly stated that the Zero flag is a Ø when set. I think that this is a perfectly logical way to expect it to operate, but in actual fact, the set or Zero state is a 1 and non-Zero is a Ø.' This is quite correct and we offer our apologies to all those people who have been struggling with the Zero flag. The above statement is true of all flags, not just the Zero flag, and we hope this error has not spoilt your enjoyment too much.

Secondly, a plea for help. Many people have written in, asking for move advanced/basic articles on Machine Code and BASIC. However, they do not tell us exactly what they want, and this is not very helpful. So if you want us to include something different please tell us. After all, WHAT we think is advanced may be considered by some as basic, and vice versa. Our other plea is for programs. We are sure that many of the people who read Print-Out have either short routines or long programs. What ever they are, what ever you do, we want to know. After all, there has never been a better time to send in your programs when you can win a copy of MAXAM on tape for your trouble. So PLEASE send your programs in.

But now onto Machine Code. If you remember, last month we printed text onto the screen, this issue we are going to print the text in different places on the screen and do one or two other things. However, for all of you who are still using the BASIC poker in the last issue, we introduce an updated version. It includes a new command, MEMORY.

MEMORY - If you wrote a machine code program that started at address &4ØØØ and then wrote a very long BASIC program, it would be possible to wipe out the machine code program. In order to prevent this you could reserve a certain amount of room for the machine code program by use of the MEMORY command. What this does is set the upper limit for a BASIC program and leaves the spare memory above this untouched for use by machine code programs. So to reserve all memory above &4ØØØ for use by machine code programs you would execute the command - MEMORY &3FFF to set the upper limit for BASIC to &3FFF. If you wish to find out where the upper limit of memory is you can use the command HIMEM - to see where the upper limit is use the instruction - PRINT HIMEM. HIMEM can also be set using the following type of expression

```
MEMORY HIMEM-1ØØ
```

For those of you have Issue 1 and the BASIC poker need to add the following line :-

```
85 MEMORY &3FFF
```

This means that none of the programs entered in print can be lost due to over-writing by a BASIC program. As far as the user is concerned the program will be how it was before. NOTE :- Any memory above HIMEM is not affected by the NEW command. For those of you who have not got the BASIC poker, the new version is printed on the next page. Version 2 of the poker appears on this issue's program cassette.

Version Two of the BASIC poker :-

```
10 REM Basic Poker2,(c) 1989 T.Defoe
20 REM Type in hexadecimal codes as shown
30 REM (Note that spaces should be disregarded
40 REM and numbers can be typed in separately)
50 REM Type 'END' to finish data entry and in
60 REM BASIC type 'CALL &4000' to make it work.
70 REM
80 REM
85 MEMORY &3FFF
90 addr=&4000
100 INPUT "What is the hexadecimal number";a$
110 IF UPPER$(a$)="END" THEN END
120 x=LEN(a$)/2
130 y=INT(LEN(a$)/2)
140 IF x=y THEN 150
145 PRINT "There is an error":GOTO 100
150 b$=LEFT$(a$,2)
160 z=VAL("&"+b$)
170 POKE addr,z
180 addr=addr+1
190 a$=RIGHT$(a$,LEN(a$)-2)
200 IF a$="" THEN GOTO 100 ELSE GOTO 150
```

If you remember, in the last issue, we printed a string on the screen using a program not unlike the one below. In this program we used labels to indicate (or stand for) various addresses and quite often these labels were preceded by a full stop (.) The program below prints out two strings - string1 and string2 - without becoming very much longer. The numbers on the right, for those who don't know, should be typed into the BASIC poker. They should not be typed into an assembler and likewise the words should not be typed into the BASIC poker.

```
ORG &4000
.txt_output equ &BB5A
ld hl.string1                    21 18 40
call print                       CD 0D 40
ld hl,string2                    21 1E 40
call print                       CD 0D 40
ret                              C9

.print
ld a,(hl)                        7E
cp 0                             FE 00
ret z                            C8
call txt_output                  CD 5A BB
inc hl                           23
jp print                         C3 0D 40

.string1
db "Hello",0                     48 65 6C 6C 6F 00
.string2
db "out there",0                 6F 75 74 20 74 68
                                 65 72 65 00
```

In this program we have used a subroutine without knowing it. In M/C you can use a subroutine exactly as you would in BASIC ie. you goto the subroutine, execute the code and then return to the main program. In the program above the subroutine is called .print. You go to it by using the command CALL print and return from it using the command RET.

You will notice that we have called the subroutine .print twice in the program and each time HL has been assigned a different value. There is one problem that will become very obvious to you when you call the program. That is you cannot print text on separate lines as you could in BASIC. Take this BASIC example below :-

```
1Ø PRINT "Hello"
2Ø PRINT "out there"
3Ø END
```

This is the BASIC equivalent of the previous program. However there is one major difference. In BASIC it is assumed that you want to print text on separate lines whereas in machine code you must tell the computer that you wish to print on separate lines. Likewise, in BASIC you must tell the computer to print on the same line - 1Ø PRINT "Hello";- whereas in machine code it is assumed.

To print text on a new line in machine code you must issue two control character commands - and they are Carriage Return (CR) and Line Feed (LF). Carriage Return moves the screen cursor to the far left-hand side of the screen but on the same line. Line Feed then moves the cursor down a line and so typing can start on the following line. The codes to print a Carriage Return and a Line Feed are 13 and 1Ø respectively. So to print the text on separate lines simply change the last four lines to read :-

```
.string1
db "Hello",13,1Ø,Ø
.string2
db "out there",13,1Ø,Ø
```

Unfortunately, it is not quite so simple to do with the BASIC poker as various addresses will have changed as well as the simple text statements. For this reason the BASIC poker numbers are written out in full at the end of this section. This program comes under the heading of printing on new lines.

Of course we are not limited to just printing on the next line, but we can print wherever we like on the screen and in any mode. The program below changes the screen mode to MODE Ø and it can be easily changed so that MODE 1 or MODE 2 can be selected.

```
.set_mode equ &BCØE
ORG &4ØØØ
LD A,Ø                          3E ØØ
CALL set_mode                   CD ØE BC
RET                             C9
```

To change modes, you load A with the value of the mode you wish to be in (either 0,1 or 2) and then call the firmware call &BCØE.The program on the next page sets the program to run in MODE 2, enables the text cursor, locates the text cursor at position 10,5 and prints the message "Hello". We have already seen the bulk of the program. The only bit which is new concerns the positioning and enabling of the text cursor and the CALLs to &BB7B, &BB6F and &BB72 take care of this. Once again the numbers for the BASIC poker are listed on the right-hand side.

```
 ORG &4ØØØ
.curs_enable equ &BB7B
.cursor_hor  equ &BB6F
.cursor_ver  equ &BB72
.txt_output  equ &BB5A
.set_mode    equ &BCØE

ld a,2                          3E Ø2
CALL set_mode                   CD ØE BC
CALL curs_enable                CD 7B BB
ld a,1Ø                         3E ØA
CALL cursor_hor                 CD 6F BB
ld a.5                          3E Ø5
CALL cursor_ver                 CD 72 BB
ld hl,message                   21 24 4Ø
CALL print                      CD 19 4Ø
RET                             C9

.print
ld a,(hl)                       7E
cp Ø                            FE ØØ
ret z                           C8
CALL txt_output                 CD 5A BB
inc hl                          23
jp print                        C3 19 4Ø

.message
db "Hello",Ø                    48 65 6C 6C 6F ØØ
```

&BB7B enables the text cursor and thus allows you to move it around the screen. The calls &BB6F and &BB72 position the screen text cursor somewhere on the screen. To set the text cursor's position you must load A with the horizontal screen position (column) and call &BB6F. The screen co-ordinate is the same as is used by the BASIC LOCATE command. In MODE Ø there are 20 characters across, in MODE 1 there are 40 and in MODE 2 there are 80. In all screen modes there are 25 rows. To set the vertical (row) co-ordinate you load A with the y value and call &BB72.

Thus in MODE 2, to.locate the text cursor in the middle of the screen you would use the following :-

```
ld a,4Ø                         3E 28
CALL &BB6F                      CD 6F BB
ld a,12                         3E 12
CALL &BB72                      CD 72 BB
ret                             C9
```

Below are printed the codes for Printing new lines :-

21 18 4Ø CD ØD 4Ø 21 2! 4Ø CD ØD 4Ø C9 7E FE ØØ C8 CD 5A BB 23 C3 ØD 4Ø

48 65 6C 6C 6F ØD ØA ØØ 65 75 74 2Ø 74 68 65 72 65 ØD ØA ØØ

In the next issue we will deal with other text calls and perhaps a bit of simple graphics. With what you've learnt a little persistance, luck and experimentation you should be able to print graphics and text in many strange and pretty ways.

If you have ever fancied programming in machine code, have read that you needed an assembler and have fainted at the price, then this competition is for you. We offer you the chance to win the very best machine code assembler, MAXAM. Not only does it assemble and disassemble machine code but also has its own built-in text editor, breakpoints and a host of other features which allow you to enjoy all the secrets of machine code the easy way.

All you have to do is write a program, in any language, which has a total length of no more than 2000 lines (in steps of 10!) and send it to us on tape together with the completed entry form below. The winner will be chosen for its originality, friendliness and the skill with which it has been programmed. Thus a 200 line masterpiece is more likely to win than a 2000 line mess. The prize is MAXAM on tape so that most people have something to gain from winning this competition and the closing date for entries is DECEMBER 20 . The winning program and the two runners-up will be printed in future issues of PRINT-OUT and to this end a listing would be appreciated but is not a necessity.

If you wish your tape to be returned please enclose 35p for postage and packing. Any entries which are not on tape will be disqualified as will entries on disc.

Entries that do not contain a completed order form will also be disqualified and people who send in more than one entry will be disqualified.

Please write your name and address on every item that you send so that nothing gets lost or damaged.

ENTRY FORM - Only ONE entry allowed per person

NAME :- ..... (Block capitals please)

ADDRESS :- .....,

.....,

.....,

.....,

POST CODE :- .....

The program ..... runs on all Amstrad CPC computers and is enclosed on tape. I give Print-out the right to publish the program should it win or be a runner-up and the program is entirely my own work.

Please sign here if you agree to the terms above :-

.....

Please tick if you include a listing :- ( )

Please send all entries to :- Thomas Defoe, PRINT-OUT, Program Comp., Bishop's Stortford, Herts CM23 2PJ.

---

10 LINE PROGRAMS

This issue contains two superb programs that were sent in by Tony Kingsmill from St. Albans.

One of them produces a rather striking graphics display - called GRID GRAFIX - and the other is a version of that old favourite, SQUASH. Both of them are written in BASIC and of course fit into those magic ten lines. Congratulations Tony! Your prize is on its way to you.

Grid Grafix

```
1Ø MODE Ø:BORDER Ø:INK Ø,Ø:RANDOMIZE TIME
2Ø FOR a=1 TO 64Ø STEP 5:MOVE a,Ø
3Ø DRAW a,4ØØ,INT(RND*16):NEXT a
4Ø FOR a=4ØØ TO 1 STEP -5:MOVE Ø,a
5Ø DRAW 64Ø,a,INT(RND*16):NEXT a
6Ø FOR x=Ø TO 15:INK x,INT(RND*27)
7Ø NEXT x
8Ø FOR y=1 TO 1ØØ
9Ø NEXT v
1ØØ GOTO 6Ø
```

Squash

```
1Ø x=1Ø:a=1Ø:b=5:i=1:k=1:sc=Ø:MODE Ø:INK Ø,1:INK 1,24
2Ø LOCATE x,22:PRINT CHR$(154):ox=x
3Ø LOCATE a,b:PRINT CHR$(111):oa=a:ob=b
4Ø IF INKEY(8)=Ø AND x>1 THEN x=x-1
5Ø IF INKEY(1)=Ø AND x<2Ø THEN x=x+1
6Ø a=a+j:b=b+k:IF a<1 THEN a=1:sc=sc+1:j=1
7Ø IF a>2Ø THEN a=2Ø:sc=sc+1:i=-1
8Ø IF b<1 THEN b=1:k=1:sc=sc+1
9Ø LOCATE ox,22:PRINT " ":LOCATE oa,ob:PRINT " ":
   IF b>21 THEN 1ØØ ELSE 2Ø
1ØØ IF a=x OR a=x-1 OR a=x+1 THEN sc=sc+2:y=22:k=-1:
   GOTO 2Ø:ELSE PRINT "SCORE :-";sc:END
```

The keys for SQUASH are the left and right cursor keys. Have fun !!!!

If you have any programs that you have written then please send them in. REMEMBER if you enter our competition you could win yourself a copy of MAXAM on tape. All you have to do is send us your program, together with the order form and of course the winner's program will be printed in Issue 3 of Print-Out. So don't delay, get your entry in today.

This page will hopefully be a permanent fixture in every magazine in the future. The idea is that you submit letters concerning a particular part of the CPC scene, and if the letters are plentiful on a subject we will group them together on this page so people can see the differing opinions of people on a certain topic. If you wish to contribute please send your letters to :- PRINT-OUT, COMMENTS PAGE, 8 MAZE GREEN ROAD, BISHOP'S STORTFORD, HERTFORDSHIRE CM23 2PJ.

SOFTWARE INDUSTRIES

In this issue we air some of your thoughts on the software industry, as well as some of our own. To start off with we have a letter from DARREN POWELL from THORNE near DONCASTER with his views on software pirates and backup copies.

I thought I might contribute one of the first letters that you will no doubt receive. It's a bit of a moan, but I hope you publish it on your letters page. The topic of my moan is the copying of software. I cannot see how software houses can justify their charges for producing a backup copy of a program you purchase from them. Also I cannot see how it can be deemed illegal to produce a single backup copy of any program that you have purchased with your hard earned cash.

In my case I always use a working copy of any program I buy and the master copy is kept in a safe place in the event of a disc corruption or mishap of some kind. Sod's Law means that this will only occur with those discs or files most highly prized or ones which are not backed up.

If the software houses were to be more helpful in the event of such mishaps perhaps the demand for 'cloners' would not be as great. As it was I think the situation did not need any alteration, the software houses were not losing vast amounts of cash through the production of backup copies and the users were happy to buy the little utility to produce backup copies for their own use. I thought that the producers of the Multiface add-on had the great sense to ensure that copied code required the presence of the Multiface to run it. Sceptics may say it merely boosts the sales of the Multiface as such users can swap programs.

My final comment and sole opinion is - make that backup copy if you dare, for your own personal use (and only of your own programs), but on no account produce numerous ones to 'swap' with friends. This sort of activity is really going to kill the software and utility scene for the CPC for good, and when you think about it, what if it was your hard work and programming that people were pirating?

We agree with all the points that you raise and have to admit to owning a Multiface II device which has recently been banned by the 'Copyright Designs and Patents Act 1988'. However, we would refute your claim that software companies did not lose vast amounts of money. I know many people who copy and swap games amongst themselves and between them, they probably lose the software companies about £300 a year and that's just three people. Of course it can be argued that the software houses are themselves directly responsible for the continued rise in the number of software pirates by charging such extortionate prices for software. Surely if all games cost no more than £4·99 then pirating software would not become nearly so profitable. Even by outlawing Multifaces and other similar copying devices it does not stop the pirates. At a guess I would say that more people own a tape-deck or hi-fi that is capable of tape-to-tape dubbing than own a software copying device. I know that copying software on a hi-fi is possible because it is how the program tapes are copied for this magazine and they work first time. However, it is undeniable that software pirates <u>are</u> killing the CPC software industry. Copying the programs you have bought for your own personal use should, we feel, be allowed but as soon as you supply it to some one else, it should become illegal. Besides, even though the new Act makes copying illegal, no-one will be caught or charged - the only way for software piracy to be eliminated is for the software companies to change their own ideas and plans as we shall see on the following page.

SOFTWARE INDUSTRIES — COMMENTS PAGE

The only way software piracy is going to be stopped is by ALL the software houses changing their minds on certain parts of their industry. Listed below are some of the things that we think need to be changed.

Many of the excellent games cost from £8·95 to £14·95 on tape and although a few do justify the price, most fall a long way short. There are also many software companies which have turned to the budget market, eg Codemasters, and they produce games which, in some cases, are better than full price games. If software piracy is to be stopped, all games need to be reduced to below £5. This would still allow budget games to be produced at £2·99 and £1·99 and would mean that it would not be worth pirating these games. Of course, the same would apply to utilities, whose price would need to be lowered from £20 to £30 down to about £15·00.

Many people feel that compilations, for one reason or another, are not worth the money. Often the compilation comes out two years after the original game was released and contain only one good game and the rest are space fillers. Sometimes the manual and instructions are poor. Others feel hard done by when they see four games that they spent £40 on appearing as a collection for under £10.

Another point which makes me, and many other users very angry, is the similarity of the games. The choice for CPC users is already very limited & when 50% are boring shoot-'em-ups and 49% are simulations that leaves only 1% of good, original games.

The final point is the cost of disc games. A tape game costs £14·99 and on disc it costs £19·99 - this means that the cost of the disc was £5. In fact, discs cost only £2·99 for us - the general public - so the software houses, who get them in bulk for under £2 make a giant profit of £3 extra for each disc game. Is it any wonder people backup tapes using the Multiface or other copiers ?!!!!!

If YOU have any comments on this, or any other topic, please write to us at the usual address. We look forward to hearing from you.

ADVENTURING — ARTICLE

In the previous issue we dealt with how adventure games are today, how they are played and what makes them enjoyable. This time we are going to look at what tomorrow's adventure games will hold.

Something that may soon become a must with all adventure games is moving graphics. Not only will we see characters moving across the screen but also performing the tasks that you command them to do - such as picking objects up. Take that one stage further and you will produce a role-playing game - such as the Bard's Tale - which doesn't have the same attraction as basic adventure games. I think that in order to constitute a proper adventure game it is necessary to type in commands rather than to have certain keys controlling certain actions. As this would ultimately lead to games requiring speed reactions as opposed to logical deduction and so destroy the main source of enjoyment for adventure games. Still the basic adventure game is getting better as software houses pack more puzzles and locations into the game as well as interactive characters. Adventuring is here to stay!!!

ADVENTURING

BITS & PIECES

In this section we will print routines and short programs, as well as tips, all of which we hope that you will find useful, or at the very least - interesting. The programs will be in any language and we'll try to cater for all tastes. So, if you have a good program (or if you need help with anything) write and tell us at the usual address.

Listed below are two routines which we hope some people will be able to incorporate into their own programs. Although they both work as stand alone units, they will be more useful when adapted as part of a much larger program. The first routine illustrates the principle behind animation graphics using two screens.

M/C

```
txt_out       equ &BB5A
set_mode      equ &BCØE
test_key      equ &BB1E
set_screen    equ &BCØ8
get_screen    equ &BCØB

org &3ØØØ                          ;program must start
                                   ;below &4ØØØ (screen)

call set_up

.loop                              ;test
ld a,47                            ;test for SPACE bar
call test_key                      ;
call nz,change                     ;if pushed, swap screens
ld a,66                            ;test for ESCAPE button
call test_key                      ;
jp z,loop                          :if not pushed then loop
ld a,&CØ                           ;otherwise, set screen to
call set_screen                    ;normal (&CØØØ) and then
ret                                :return to BASIC.

.change
call get_screen                    :get screen address
cp &CØ                             ;if it is &CØØØ then
jp z,forty                         :goto subroutine
ld a,&CØ                           ;otherwise set to &CØØØ
.dummy
call set_screen                    ;set the screen address
ret                                ;return from subroutine

.forty
ld a,&4Ø                           ;ld a with screen address
jp dummy                           ;and return.

.set_up
ld a,&CØ                          ;set screen address to &CØØØ
call set_screen                    ;and change it
ld a,1                             ;set the screen mode to
call set_mode                      ;one (MODE 1)
ld b,24Ø                           ;b=240
ld a,"1"                           ;a=ASCII code of number 1
.loop2
call txt_out                       ;print "1"
djnz loop2                         ;and do it 240 times
ld a,&4Ø                           :set screen address to &4ØØØ
call set_screen                    ;and change it
```

(continued on next page)

(continued from previous page)

```
ld a,1                          ;set the screen mode to
call set_mode                   :one (MODE 1)
ld b,24Ø                        ;b=240
ld a,"2"                        ;a=ASCII code of number 2
.loop3
call txt_out                    ;print "2"
djnz loop3                      ;and do it 240 times
ld a,&CØ                        :set screen address to &CØØØ
call set_screen                 ;and change it
ret                             ;return from subroutine
```

What the program does is set up two screens whose addresses start at &4000 and &C000. The first of these is filled with six lines of twos and the second with six lines of ones. When you press the SPACE bar, the program switches rapidly between the two screens, and so you see a blur of ones and twos. The idea is that whilst screen one is shown, screen 2 is updated and then the screens are swapped over. This process is repeated to produce flicker-free animation. The only real disadvantage with this method is that the two screens take up half the computer's memory. The program is written using jumpblock calls and so you should be able to look them up without too much trouble. Below is the BASIC poker version, so that even if you don't have an assembler you can still run the program.

NB :- The screen memory is usually located at &CØØØ.

BASIC

```
 1Ø addr=&3ØØØ
 2Ø READ a$
 3Ø IF a$="END" THEN GOTO 8Ø
 4Ø a=VAL("&"+a$)
 5Ø POKE addr,a
 6Ø addr=addr+1
 7Ø GOTO 2Ø
 8Ø CALL &3ØØØ
 9Ø END
1ØØ DATA CD,2C,3Ø,3E,2F,CD,1E,BB
11Ø DATA C4,19,3Ø,3E,42,CD,1E,BB
12Ø DATA CA,Ø3,3Ø,3E,CØ,CD,Ø8,BC
13Ø DATA C9,CD,ØB,BC,FE,CØ,CA,27
14Ø DATA 3Ø,3E,CØ,CD,Ø8,BC,C9,3E
15Ø DATA 4Ø,C3,23,3Ø,3E,CØ,CD,Ø8
16Ø DATA BC,3E,Ø1,CD,ØE,BC,Ø6,FØ
17Ø DATA 3E,31,CD,5A,BB,1Ø,FB,3E
18Ø DATA 4Ø,CD,Ø8,BC,3E,Ø1,CD,ØE
19Ø DATA BC,Ø6,FØ,3E,32,CD,5A,BB
2ØØ DATA 1Ø,FB,3E,CØ,CD,Ø8,BC,C9
21Ø DATA END
```

We have found that an excellent way of converting machine code that has been assembled by an assembler, into DATA lines, is to enter the short line below. It prints out the data complete with commas and all you need to do is put DATA commands and line numbers at the beginning. You must put the correct start and end addresses in the line below:-

```
FOR i=startaddr TO endaddr:PRINT HEX$(PEEK(i));:PRINT ",":NEXT i
```

The second routine is a four direction scrolling routine. It uses only jumpblock firmware calls and so is a lot less smooth than it could be. Listed below is the machine code program :-

```
get_scr_addr        equ &BCØB
set_scr_addr        equ &BCØ5
get_key_wait        equ &BB1E
wait_flyback        equ &BD19
hard_scroll         equ &BC4D
ink_encode          equ &BC2C

org &4ØØØ

.loop
ld a,8                          ;test for left cursor
call get_key_wait               ;key and if pressed
call nz,left                    ;the CALL .left
ld a,1                          ;test for right cursor
call get_key_wait               ;key and if pressed
call nz,right                   ;then CALL .right
ld a,Ø                          ;test for up cursor
call get_key_wait               ;key and if pressed
call nz,up                      ;then CALL .up
ld a,2                          ;test for down cursor
call get_key_wait               ;key and if pressed
call nz,down                    ;then CALL .down
ld a,66                         ;test for ESCAPE key
call get_key_wait               ;and if not pressed
jp z,loop                       ;then jump to loop
ret                             ;otherwise RETURN

.left
call get_scr_addr               ;get current screen address
inc hl                          ;and change it so
inc hl                          ;the screen moves left
call set_scr_addr               ;and tell the computer
call wait_flyback               ;wait for CRT
ret                             ;and return

.right
call get_scr_addr               ;get current screen address
dec hl                          ;and change it so
dec hl                          ;the screen moves right
call set_scr_addr               ;and tell the computer
call wait_flyback               :wait for CRT
ret                             ;and return

.up
ld a,Ø                          ;set the colour of the new
call ink_encode                 ;line and encode it.
ld b,1                          ;set it to scroll up
call hard_scroll                ;and tell computer
ret                             ;return

.down
ld a,Ø                          ;set the colour of the new
call ink_encode                 ;line and encode it
ld b,Ø                          ;set it to scroll down
call hard_scroll                ;and tell computer
ret                             ;return
```

The main loop of the program tests for the various cursor keys and the ESCAPE key. When any of these are pressed the program then executes one of the subroutines to perform the task.

The move left and right subroutines take the screen offset and modify it. When this is done the modified offset is sent out to the computer which changes it.

The move up and down routines use the hardware scroll. To scroll up b must contain any non-zero number and the colour of the next line must also be specified. This is an encoded ink and so the firmware call &BC2C must be used to give us the mask.

You may have noticed that the left and right routines have a CALL wait_flyback instruction in them whilst the up and down ones do not. The reason is that the left and right ones need to wait for the CRT otherwise a blur is produced and the other two do not because they are slow already.

There are two problems with this program :-

a) It is slow and not incredibly smooth - this is because it uses the firmware calls.

b) Anything that passes off the top of the screen is lost and when scrolling horizontally the bottom line (or top line) reappears at the top (or bottom) of the screen when it is scrolled off. Both problems can be solved by incorporating it in a larger and more complex program.

Below is shown the BASIC poker version - before you start asking, yes, line 11Ø is meant to be shorter than the others.

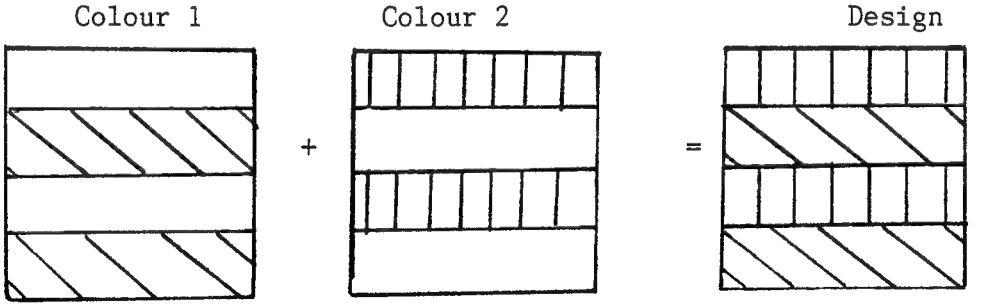
```
 1Ø addr=&4ØØØ
 2Ø READ a$
 3Ø IF a$="END" THEN GOTO 8Ø
 4Ø a=VAL("&"+a$)
 5Ø POKE addr,a
 6Ø addr=addr+1
 7Ø GOTO 2Ø
 8Ø CALL &4ØØØ
 9Ø END
1ØØ DATA 3E,Ø8,CD,1E,BB,C4,29,4Ø
11Ø DATA 3E,Ø1,CD,1E,BB,C4,35
12Ø DATA 4Ø,3E,ØØ,CD,1E,BB,C4,41
13Ø DATA 4Ø,3E,Ø2,CD,1E,BB,C4,4C
14Ø DATA 4Ø,3E,42,CD,1E,BB,CA,ØØ
15Ø DATA 4Ø,C9,CD,ØB,BC,23,23,CD
16Ø DATA Ø5,BC,CD,19,BD,C9,CD,ØB
17Ø DATA BC,2B,2B,CD,Ø5,BC,CD,19
18Ø DATA BD,C9,3E,ØØ,CD,2C,BC,Ø6
19Ø DATA Ø1,CD,4D,BC,c9,3E,ØØ,CD
2ØØ DATA 2C,BC,Ø6,ØØ,CD,4D,BC,C9
21Ø END
```

That concludes this issue's section on tips and routines and we hope that you have found it interesting. REMEMBER if you have something you want to know, or have something you want other people to know, then please write to us at the normal address.

Multi-Colour Sprites

Many professional programs use multi-coloured sprites and it is possible to produce these from BASIC with a little care. The secret lies with the SYMBOL and CHR$ commands.

First of all you need to produce and design the character or sprite and to do this it is advisable to use squared paper (or the Character Designer program in this issue). You need to design a symbol for each colour in the character so that when they are put together they mesh perfectly. This is shown below :-



The character must be designed on an 8 x 8 grid, although for larger characters more than one symbol can be joined together. In the example above, SYMBOL 254 (Colour 1) can be recorded as follows - SYMBOL 254,Ø,Ø,255,255,Ø,Ø,255,255 - and SYMBOL 255 (Colour 2) as - SYMBOL 255,255,255,Ø,Ø,255,255,Ø,Ø. Before they can be redefined you must issue SYMBOL AFTER 32 command which allows all symbols to be redefined. Thus, our program reads so far :-

```
1Ø SYMBOL AFTER 32
2Ø SYMBOL 254,Ø,Ø,255,255,Ø,Ø,255,255
3Ø SYMBOL 255,255,255,Ø,Ø,255,255,Ø,Ø
```

However, if we print these characters one over the other, the lower character will be wiped out by the second, so we must issue the PRINT CHR$(22)+CHR$(1) command after any MODE instructions have been issued. So our program continues :-

```
4Ø MODE Ø
5Ø PRINT CHR$(22)+CHR$(1)
```

Now it just remains for us to print the characters and change the inks and pens accordingly. Thus our program continues :-

```
6Ø INK 1,24:INK 2,7
7Ø LOCATE 1Ø,12:PEN 1:PRINT CHR$(254)
8Ø LOCATE 1Ø,12:PEN 2:PRINT CHR$(255)
```

This does what we want but the last two lines are going to be very time consuming if we have to print 100 of these designs on the screen. Luckily there is a better way and it uses more CHR$ commands. We have already used CHR$(22)+CHR$(1) which sets up a transparent text option and CHR$(22)+CHR$(Ø) turns it back to normal. To change pens you use CHR$(15)+CHR$(n) where n is the value of the pen you want to use. And instead of using two locate commands we can use CHR$(8) which tells the text cursor to move back one space. So to print this character we would use - PRINT CHR$(15)+CHR$(1)+CHR$(254)+CHR$(8)+CHR$(15)+CHR$(2)+CHR$(255). If we assign a variable to this, eg CHAR$, then we can print the design very simply by using one locate and then PRINT CHAR$. So change lines 7Ø and 8Ø to :-

```
7Ø char$=CHR$(15)+CHR$(1)+CHR$(254)+CHR$(8)+
   CHR$(15)+CHR$(2)+CHR$(255)
8Ø LOCATE 1Ø,12:PRINT char$
```

The program below creates a large multi-coloured MODE Ø space-ship which could be used as scenery for part of a game and is printed using the command - PRINT rocket$;rocket2$;rocket3$

```
 1Ø SYMBOL AFTER 32
 2Ø SYMBOL 22Ø,1,3,7,15,31,63,12Ø,24Ø
 3Ø SYMBOL 221,195,195,231,255,255,255,1Ø2,36
 4Ø SYMBOL 222,128,192,224,24Ø,248,252,3Ø,15
 5Ø SYMBOL 223,6Ø,6Ø,24,Ø,Ø,Ø,Ø,Ø
 6Ø SYMBOL 224,Ø,Ø,Ø,Ø,255,255,255,255
 7Ø SYMBOL 225,255,255,255,255,Ø,Ø,Ø,Ø
 8Ø SYMBOL 226,28,6Ø,124,1Ø8,12,12,126,126
 9Ø SYMBOL 227,Ø,Ø,Ø,Ø,243,243,129,129
1ØØ SYMBOL 228,227,195,131,147,Ø,Ø,Ø,Ø
11Ø SYMBOL 229,Ø,Ø,6Ø,6Ø,Ø,Ø,255,255
12Ø SYMBOL 23Ø,Ø,24,Ø,Ø,126,126,Ø,Ø
13Ø SYMBOL 231,Ø,Ø,1,3,Ø,Ø,Ø,Ø
14Ø SYMBOL 232,Ø,Ø,Ø,Ø,7,7,15,31
15Ø SYMBOL 233,63,63,127,127,Ø,Ø,Ø,Ø
16Ø SYMBOL 234,Ø,Ø,Ø,Ø,255,252,248,24Ø
17Ø SYMBOL 235,Ø,Ø,128,192,Ø,Ø,Ø,Ø
18Ø SYMBOL 236,Ø,Ø,Ø,Ø,224,224,24Ø,248
19Ø SYMBOL 237,252,252,254,254,Ø,Ø,Ø,Ø
2ØØ SYMBOL 238,Ø,Ø,Ø,Ø,255,63,31,15
21Ø rocket$=CHR$(22Ø)+CHR$(221)+CHR$(222)+CHR$(8)+CHR$(8)+
    CHR$(15)+CHR$(3)+CHR$(223)+CHR$(8)+CHR$(11)+CHR$(224)+
    CHR$(8)+CHR$(15)+CHR$(1)+CHR$(225)+CHR$(8)+CHR$(11)+CH
    R$(8)+CHR$(233)+CHR$(228)+CHR$(237)+CHR$(8)+CHR$(8)+CH
    R$(8)+CHR$(15)+CHR$(3)+CHR$(234)
22Ø rocket2$=CHR$(11)+CHR$(9)+CHR$(227)+CHR$(238)+CHR$(8)+
    CHR$(8)+CHR$(15)+CHR$(15)+CHR$(226)+CHR$(8)+CHR$(11)+C
    HR$(8)+CHR$(15)+CHR$(3)+CHR$(232)+CHR$(224)+CHR$(236)+
    CHR$(8)+CHR$(8)+CHR$(8)+CHR$(15)+CHR$(1)+CHR$(231)+CHR
    $(225)+CHR$(235)+CHR$(11)
23Ø rocket3$=CHR$(9)+CHR$(11)+CHR$(225)+CHR$(8)+CHR$(15)+C
    HR$(3)+CHR$(224)+CHR$(8)+CHR$(11)+CHR$(15)+CHR$(1)+CHR
    $(225)+CHR$(8)+CHR$(15)+CHR$(3)+CHR$(224)+CHR$(8)+CHR$
    (11)+CHR$(15)+CHR$(1)+CHR$(23Ø)+CHR$(8)+CHR$(15)+CHR$(
    3)+CHR$(229)
24Ø MODE Ø:PRINT CHR$(22)+CHR$(1)
25Ø LOCATE 1,1Ø:PRINT rocket$;rocket2$;rocket3$
26Ø GOTO 26Ø
```

In the above listing, lines 1Ø - 2ØØ set up all the characters and lines 21Ø - 23Ø put them all together. In these lines you may have noticed CHR$(8), CHR$(9), CHR$(1Ø) and CHR$(11) and these commands move the graphics cursor around and their functions are shown below. As there is a limit of 255 characters to a BASIC line I could not put all of the CHR$ commands in one variable but had to split them between three. Line 25Ø pulls these all together in the write order. Line 26Ø creates an infinite loop so that the Ready message doesn't spoil the display.

CHR$(8) - Back one character
CHR$(9) - Forward one character
CHR$(1Ø) - Down one line
CHR$(11) - Up one line

These control character commands are all listed in your computer's manual and are very useful for writing quick and shortish programs.

HOME→BREW

Often homebrew software is well written, comprehensive and, what is more, it is useful. The serious market for the CPC is notoriously poor and so homebrew software has flourished. There are many small 'companies' all offering something new & different. This page will be set aside to review these products.

In this issue, we turn our attention to a small software company, MiP Software, that is run by Matthew Pinder and which has just released a new program for the CPC called Maths Master +.

Math's Master Plus is a follow up to MiP's earlier program, Maths Master, and this new version boasts over 140 formulae and conversions. These functions are all accessed through several easy-to-use menus which are very clear and helpful. All the functions are listed on a well produced A4 instruction booklet which tells you how to get to the required section in Math's Master Plus. This program is not designed to help you learn maths in any way but to simply provide you with one of the most sophisticated calculators. This is why the instruction booklet does not give you any more information about the functions as it presumes that if you want to use a function you will know about it. But the instruction booklet does include diagrams for use with SIN, COS and TAN. The program is clear, simple and helpful and although it does not have pretty screen layouts and multi-colour numbers it is no worse for it. After all, as you are waiting for the answer to ax+bx+c you don't want to have the letters and numbers tumbling about the screen until they form a pattern and all the colours flashing wildly - what you want is the answer and this is what Math's Master Plus gives you. When I say that the program does not have pretty screen layouts this does not mean that it is an unorganised mess - it is not. It is what you might call a functional program - no frills just neat, tidy and accurate. One thing that seemed to be missing were simple equations such as 45+b=35 - I now this is a very simple example but for more complicated formula and equations it might have been helpful. Having said that, there seemed to be virtually everything else. The only major problem with the program is the length of time that it takes to load on tape. If you just want to answer one simple question it is very inconvenient to have to wait over 10 minutes for it to load. Of course if you had many similar questions to answer then it is ideal - many a student will find it invaluable. On disc of course it is a different matter - if you have a disc drive dont't hesitate to buy it! Even on tape I would still recommend that you buy it as you never know when you may need it. Ideally of course would be to transfer it ROM and have a fully fledged calculator instantly available. Overall it is an excellent buy and at only £3·95 (tape) or £6·95 (disc) it is a bargain and if you use the order form below you will save yourself £1 off either version.

Another, older program which MiP market is Sharewatcher - a simple stockmarket simulation on the CPC. This great fun for anyone with ambitions to make money (and lots of it!). It is good fun to play and will keep you amused for upto ½ an hour a game. We look forward to the more complicated version later this year. Price - £2·50 (tape) with tape-disc back-up routine.

Please send to :- Matthew Pinder, MiP, 4 Wham Hey, New Longton, Preston, LANCS.

ITEM	TAPE	DISC
Math's Master Plus	£2·95 ( )	£5·95 ( )
Sharewatcher	£2·50 ( )	---

NAME :- .....,
ADDRES :- .....,
.....,
.....,
POST CODE :- .....

PLEASE MAKE CHEQUES/POSTAL ORDERS PAYABLE TO M. PINDER (All prices include P+P).

PRINT-OUT ISSUE II

Palette Swapping ANIMATION

One of the cleverest uses of BASIC colours is to create the illusion of animation by the use of palette swapping. Although it can only be used for simple movement it is still very effective and simple to set up.

The theory behind palette swapping is that you draw the lines or shapes in the positions you want them to move to, but each line is in a different ink. Therefore, the number of lines you can have is limited by the number of different inks available to you in a certain mode. One way around this is to repeat the inks after a certain number of lines. This means that instead of having just one line or shape on the screen at any one time, you can have as many as you like. The program below sets up the screen and then animates a series of lines that go across the screen.

```
 1 REM Palette swapper (c) 1989 T.Defoe
10 MODE 0
20 CALL &BC02
30 k=1
40 FOR i=0 TO 640 STEP 4
50 FOR j=1 TO 4
60 MOVE (i-1)+j,1
70 DRAW (i-1)+j,400,k
80 NEXT j
90 k=k+1:IF k=14 THEN k=1
100 NEXT i
110 FOR i=1 TO 14
120 INK i.0
130 NEXT i
140 FOR i=1 TO 14
150 INK i,26:CALL &BD19
160 INK i,0:NEXT i
170 GOTO 140
```

Lines 10 - 30 set up the screen mode, reset the colours and does all the initialisation. The firmware call, &BC02, resets the colours amongst other things and so the program will run correctly if you run it a second time. Lines 40 - 100 draw the multi-colour bands on the screen and the ink to be used is stored in 'k' and is modified in line 90. Lines 110 - 130 set all the inks to 0 (Black). Lines 140 - 170 actually do the animation. What happens is that each ink is turned to white, the computer waits for a 'Frame Flyback' (CALL &BD19) and then turns the ink back to black. When all 14 inks have been flashed the cycle is repeated. CALL &BD19 can be replaced by the 'FRAME' command and all it does is reduce flicker. Try the program without it and spot the difference !

The next program shows that not just lines, but shapes can be used in this way.

```
10 REM Shape Shifter
20 MODE 0
30 CALL &BC02
40 BORDER 7:INK 0,0
50 k=1
60 FOR i=1 TO 180 STEP 18
70 MOVE 10+i,10+i
```

(continued on next page)

(continued from previous page)

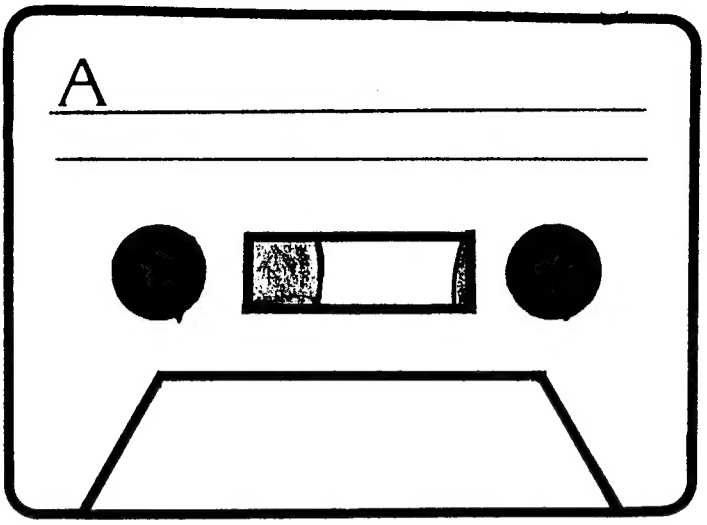
```
 8Ø DRAW 63Ø-i,1Ø+i,k
 9Ø DRAW 63Ø-i,39Ø-i
1ØØ DRAW 1Ø+i,39Ø-i
11Ø DRAW 1Ø+i,1Ø+i
12Ø k=k+1
13Ø NEXT i
14Ø FOR i=1 TO 1Ø
15Ø INK i,Ø
16Ø NEXT i
17Ø FOR i=1 TO 1Ø
18Ø INK i,26:CALL &BD19
19Ø INK i,Ø:FOR t=1 TO 5Ø:NEXT t:NEXT i
2ØØ FOR i=1Ø TO 1 STEP -1
21Ø INK i,Ø:CALL &BD19
22Ø INK i,26:FOR t=1 TO 5Ø:NEXT t:NEXT i
23Ø GOTO 17Ø
```

Of course we can do a similar thing with text and this next program moves an arrow, CHR$(243), along the top of the screen. This is interrupt driven so that the program can get on with the rest of it without having to worry about moving an arrow. The rest of the program is simulated by line 2ØØ which can be replaced with your own program.

```
 1Ø REM Arrow Mover
 2Ø MODE Ø:CALL &BCØ2
 3Ø WINDOW #1,1,2Ø,1,2
 4Ø CLS #1
 5Ø WINDOW #Ø,1,2Ø,3,25
 6Ø INK Ø,Ø:CLS #1
 7Ø k=1:BORDER 7
 8Ø LOCATE #1,3,1
 9Ø FOR i=1 TO 14
1ØØ PEN #1,k
11Ø PRINT  1,CHR$(243);
12Ø k=k+1
13Ø NEXT i
14Ø FOR i=1 TO 14
15Ø INK i,Ø
16Ø NEXT i
17Ø INK 15,26:PEN 15
18Ø i=1
19Ø EVERY 1 GOSUB 1ØØØ
2ØØ GOTO 2ØØ
21Ø END

1ØØØ INK i,26:CALL &BD19:INK i,Ø
1Ø1Ø i=i+1:IF i=15 THEN i=1
1Ø2Ø RETURN
```

The interrupt EVERY is used in line 19Ø to control the movement of the arrow and the subroutine from 1ØØØ - 1Ø2Ø actually moves it. To slow the arrow down change line 19Ø to read - EVERY n GOSUB 1ØØØ where 'n' is the time delay in 1/50 of a second.



OFFERS

Please make all cheques/postal orders payable to T.J. Defoe. All orders will be dealt with on a first-come first-served basis, although programs and back issues can be guaranteed. Please do NOT send cash unless it cannot be avoided. Please send any orders to :- PRINT-OUT, Special Offers, 8 Maze Green Road, Bishop's Stortford, Hertfordshire CM23 2PJ.

Cassettes

If you wish to receive a tape containing all the programs in this issue and a booklet that explains how they and many others work, please send either :- a) A blank tape (12 min) + 50p (p+p)
or b) £1·00 (which includes tape and p+p)
The program cassette for Issue 1 is still available at the above prices. PLEASE not that Character Designer appears on both tapes. Please mark your order clearly, Issue 1 or Issue 2.

Back Issues

Have you just bought Issue 2 of Print-Out and want to find out what Issue 1 was like ? Well, you can obtain a copy of Issue 1 by sending 70p and a large A4 Size S.A.E. (26p stamp) to the usual address.

Special Offer

In this issue you have the chance to buy two pieces of excellent homebrew software written by Michael Pinder. They are Math's Master Plus - £2·95 (tape)/£5·95 (disc) - at a £1 discount and Sharewatcher - £2·50 (tape). You will already have seen the reviews on page and we are sure you will agree that they are both superb pieces of software. Please use the order form on page 37 to order and SAVE £1. All cheques/postal orders for this offer should be made payable to M. Pinder and sent to :- Matthew Pinder, MiP, 4 Wham Hey, New Longton, Preston, LANCS.

Also, don't miss your chance to win MAXAM, the machine code assembler for the Amstrad CPC, on page 27 .

Small Ads.

FOR SALE: Games (arcade, strategy & adventure) and Serious Software. Tapes and Discs. All originals. Low Prices. For list send SAE to 29 St Leonards Road, East Sheen, London SW14 7LY or phone 01-876-5245.

User Clubs

UNITED AMSTRAD USER GROUP: Write to United Amstrad User Group, The Chairman, 26 Uplands Crescent, Fareham, Hants, PO16 7JY for information on how to join. The cost for a year's membership is £7.00 (UK), £10.00 (Europe) and £14.00 (overseas).

Remember you can place an advertisment of upto 40 words (including name and address) in this section of the magazine, free of charge.

Issue 3

If you would like to contribute to Print-Out in any way, then please get in touch at our address which is :- PRINT-OUT, 8 Maze Green Road, Bishop's Stortford, Hertfordshire CM23 2PJ. If you've written a program, discovered something new, done something interesting or have a problem with the Amstrad CPC then we would love to hear from you. Any articles, programs, etc. that you would like considered for publication in Issue Three, should arrive no later than the 20th December. (Please mark them clearly 'PRINT-OUT' - otherwise they may get stuck under the Christmas Tree !!!)

Thank you.

Thomas Defoe, Mark Gearing and Jonathan Haddock.

PRINT-OUT, 8 Maze Green Road, Bishop's Stortford,
Hertfordshire CM23 2PJ

Unfortunately, there were quite a few errors in programs for Issue Two and the corrections are shown below. We are very sorry for any inconveniance that these errors may have caused and have improved our program checking method from Issue Three onwards.

CHARACTER DESIGNER :- (pages 19-21)

```
 90 WINDOW #1,2,9,2,9
860 IF TEST(60+(2*x),100-(2*y))<>1 THEN 840
940 WHILE INKEY(47)<>-1:WEND
```

NAME AND ADDRESS STORER :- (pages 8-12)

```
230 where=INSTR(know$,a$)
300 MOVE 0,0:DRAW 0,399:DRAW 639,399:DRAW 639,0:DRAW 0,0
310 WINDOW #0,2,79,4,20:WINDOW #1,2,79,21,24:PEN #1,0:PAPER #1,1
340 DIM name$(1000):DIM phone(1000):DIM address$(1000,4)
830 NEXT j
870 LOCATE 20,14:INPUT "Enter correct name :- ",name$(i)
1080 NEXT j
1170 LOCATE 20,5:PRINT "Number :- ";phone(i)
1510 FOR i=1 TO number
1620 IF UPPER$(address$(i,j))=UPPER$(find$) THEN RETURN
1630 NEXT i
1780 INPUT address$(number,i)
1980 j=1
2010 b$=MID$(name$(two),j,1)
2100 hold$=name$(one):name$(one)=name$(two):name$(two)=hold$
```

MACHINE CODE PROGRAMMING :- (page 26)

The codes for printing new lines should read as follows :-

```
21 18 40 CD 0D 40 21 20 40 CD 0D 40 C9 7E FE 00 C8 CD 5A BB 23 C3 0D 40
48 65 6C 6C 6F 0D 0A 00 6F 75 74 20 74 68 65 72 65 0D 0A 00
```